

REDACTED

Therefore **REDACTED** for the **inner circle** to use **REDACTED**
or TYB production.

(4) V.1.22 **REDACTED** : **REDACTED**
1.21. Then

Therefore **REDACTED** for the **inner circle** to use **REDACTED**
for The Yield Book production. See the color charts, for the comparison
of ACE Theft Code installed in brown, before they were converted to ACE Use Code, and
new ACE Use Code marked in grey, 1998-7-30.png, Ex. CCC, in connection to
REDACTED

180. In MR-RCS, CGM00176, there is no definition for the functions
`yb_get_term_struct()` or `yb_save_term_struct()`. Defendants repeatedly
reassured Plaintiff that MR-RCS is the complete set of RCS code, at least regarding to term
structures. Clearly without the definitions of these functions, none of The Yield Book system
compiled at Mortgage Research would have run since July 1998. RCS-MR should have the code
files containing the definitions of `yb_get_term_struct()` and
`yb_save_term_struct()`. Accordingly, it appears that Defendants have withheld certain
crucial code files in MR-RCS from discovery, along with the other **offline code** which reads in
ACE sequences.

181. There is no mystery about what the functions `yb_get_term_struct()` and
`yb_save_term_struct()` do. *This ACE Use Code also made its first appearance in a new
suite of code files in ACE Use Code installed on June 16, 1998 in TYB RCS code, which was less
two weeks after ACE theft.* These functions appeared in a new code file `cmooas_hashts.c,v`
(Ex. CCCCC) which made the first appearance on June 16, 1998 in TYB-RCS (CGM00177),
and was part of the ACE Use Code. I would expect that Defendants might need one to two weeks
to reverse the ACE sequences from the stolen ACE term structures. So it appears that Defendants
put the stolen ACE sequences into production as soon as they backed out the ACE sequences.

182. The Yield Book RCS code written by The Yield Book Director of Research, Dr. Stewart Herman, was added to the Mortgage Research RCS as indicated by the entry ^{REDACTED} Ex. EEE on p. 19). Apparently this was part of an effort to speed up the process to get ACE into The Yield Book production as soon as possible. Plaintiff has requested The Yield Book RCS code, but Defendants successfully resisted producing even though they would have incurred little burden beyond providing a disk.

183. There are many commands and procedures for Theft Code,

^{REDACTED} *multiple* sets of term structures in order to back out⁵⁶ one single ACE sequence. After the theft was completed, this code was upgraded to ACE Use Code, supplemented by the ^{REDACTED} version of these functions,

REDACTED

see Fig. 2, 3, and 6 of the flow charts, Ex. SSS. The previous functions that gave the **inner circle**, **REDACTED** access to both the ACE Interest Rate Paths and interest rate model inputs necessary to back out ACE are all carefully blocked off from the **online code** users who can now access the ACE Interest Rate Paths. Yet these blocked off functions that allow access to the information necessary to back out ACE remain accessible to Defendants' **inner circle** who still can run these **REDACTED**

184. For more than three months from the installation of V1.21 of April 8, 1998 to V1.22 of July 9, 1998, of `pathgen_cover.c`, v, there were no revisions. See Ex. EEE log dates. However, after the ACE theft had served its purpose of stealing ACE and once Theft Code was being revised to be transformed into "production" strength code by the ACE Use Code, there was a flurry of frantic activities within two weeks as describe below. In fact, there were four major revisions to `pathgen_cover.c` alone, within just 3 weeks:

⁵⁶For example, ^{REDACTED}

1. V1.22 of July 9, 1998, the login comments state: "@alter #ifdef YB code", and, as shown above, the two functions "yb_get_term_struct()", "yb_save_term_struct," introduced on June 16, 1998 in TYB-RCS code file cmooas_hashts.c,v (Ex. CCCCC), were added to the new version.
2. In V1.23 of July 23, 1998 in pathgen_cover.c,v (Ex. EEE), the login comment reads: "@add calls to use indexed path files", and four functions "fetch_xfiles, save_xfiles, read_paths, write_paths" were added to the new version. And the code file xfile.c (See Ex. IIII) was added to the ACE Use Code for the **inner circle** people to use to use the ACE sequence to generate ACE term structures (stored in the object PathMem, under the pointer &pm) using the **offline** ACE Use Code. See below for further explanations of the xfile.c.
3. In V1.24 of July 24, 1998 the login comment reads: "@hide PathID from Yield Book". This shows that Yield Book **online** users, including their own traders and employees, except the **inner circle** people, will only have access to the contents of PathMem, not PathID which contains all the information of model parameters, market curves etc, sufficient to back out ACE sequences from ACE term structure PathMem.
4. In V1.25 of July 30, 1998, in pathgen_cover.c,v (Ex. EEE) the login comment reads: "@save libor perturbation curve in ID".

185. All the code and comments referenced in the previous paragraph were created by Mr. Russell, who appears to have been so busy, that he forgot to clean up obsolete code ^{REDACTED} in V1.22. This was uncharacteristic for meticulous programmer like Mr. Russell.

186. Through May 2005, the date when RCS was produced, the last version of **REDACTED** V1.38, Ex. WWW and EEE, p.1, still shows the continued use of ACE in TYB production as indicated from V1.22 above (Ex. VVV). It was moved from development into production. Defendants represented that through July 2007, no material changes were made to the sequences Defendants used (Radak Decl. p.2 Ex. O) If that is true, and I have not seen any evidence to show that this was not the case, then the use of ACE has continued through at least July 2007.

187. In v1.21, there are conditions placed on Theft Code in **REDACTED** : that allows it to be accessed only for the **inner circle** people to use, i.e., then by the Mortgage Research Department people who run the ACE tests:

Near the bottom of P.6, Ex. DDD:

^{REDACTED}

REDACTED

Near the end of P.7, Ex. DDD:

REDACTED

*nspaths, *npaths, dyc,

188. In V1.22 (Ex. VVV) after the theft of ACE, as noted above, both restrictions `#ifndef YB`⁵⁷ are removed, because the ACE Theft Code was revised to function as the ACE Use Code for The Yield Book production. Thus, at first the **offline code** calls `save_paths()` to save ACE term structures. Then the Yield Book **online code** will call

```
if (!get_paths(*noutcrv, time_dim, *nspaths, *npaths, dyc,
paryc)) { /* returns TRUE if we can get paths */
    status = PATHGEN(...)
```

to get the ACE term structures saved **offline** for **online** use.

189. C programming rules require the computer to check the criterion inside `()` after “if” to see if it is zero (FALSE) or nonzero (TRUE). If zero, the computer will skip the code in the brackets `{}` following the “if `()`”. And if nonzero, the computer will go into the `{}` and run the code in the `{}`. Only if `get_paths()` fails to return the value zero will the computer go into the `{}` and run `pathgen()`, Defendants’ pre-ACE interest rate paths generation code. See the definition of `get_paths()` in `pathgen_cover.c,v` (Ex. EEE, p.7):

```
static Boolean58 /* TRUE if paths obtained here */
```

See also Figs. 3 and 6, Ex. SSS. Therefore, after `pathgen_cover.c` V1.22 on July 9, 1998, which was shortly after the theft of ACE, Defendants’ registered clients of The Yield Book, including their own traders, relied on the stored ACE term structures for production. Only when `get_paths()` fails to get the ACE term structures generated using an ACE sequence would it run the function `pathgen()` which would then use Defendants’ **online code** sequence generator

⁵⁷ As shown previously, “ifndef YB” means not for Yield Book online use, only for the **inner circle** show can access the **offline code**.

⁵⁸ “Boolean” is a data type that can only assume one of two values (true or false). http://en.wikipedia.org/wiki/Boolean_data_type.⁵⁹ To put this step in perspective with respect to the entire The Yield Book valuation process, once the Cash Flow, the expected interest and principle payments from the mortgages from the prepayment model which generated the cash flow using the set of interest rate paths generated by the sequence, is discounted by those same interest rates whose accuracy depends on the sequences, the cash flow generated by all interest rate paths is averaged together to get the value of the fixed income security.

to obtain interest rate paths.

190. To summarize, similar to **offline** Theft Code, the **offline** ACE Use Code is also running hidden from The Yield Book **online code** users, and Plaintiff in discovery, see Table B. Compare the hidden dotted line code and files in Fig.2 and Fig. 3, Ex. SSS. The flow chart in Ex. CCC illustrated **REDACTED** by the ACE Use Code produced by Defendants: **REDACTED**

191. In broad brush, the **offline** ACE Use Code is designed to read-in ACE sequences from the ACE data file (which is hidden from The Yield Book **online** users and was not produced in discovery) (Fig. 3, Ex. SSS, "ACE Seq. Paths"). The **offline** ACE Use Code uses ACE to generate ACE Interest Rate Paths in the **offline** code (not accessible by **online** The Yield Book users) see Fig.3, Ex. SSS, **REDACTED** Then the **offline** ACE Use Code will **REDACTED** for The Yield Book production to "save" ACE Interest Rate Paths into hidden files.

192. Then to allow Defendants' **internal users**, including their own traders, to use the ACE Interest Rate Paths, the **online** ACE Use Code **REDACTED** is used to find the hidden directory where the ACE Interest Rate Paths are stored, and "get" the stored ACE Interest Rate Paths to use. This is shown in Fig.3, Ex. SSS where the **REDACTED** (containing the ACE Interest Rate Paths). **REDACTED** The ACE Interest Rate Paths are used to generate cash flow in the "Cash Flow Generator" and to discount the "Cash Flow" as shown in Fig.3, Ex. SSS.⁵⁹

193. When Defendants installed ACE Use Code on June 16, 1998 (in **REDACTED** Ex. CCCCC), they **REDACTED** on ACE Theft Code performing functions to ACE Use Code performing (b2) and (b3), in Table B, see the Top Level Description. The offline ACE Use Code also uses the function (b1) in Table B which is similar to (a1) in Table A. **REDACTED**

The code to read in ACE sequences is very small⁶⁰. The offline ACE Use Code could even be the same as the online ACE Use Code **REDACTED** Defendants withheld or destroyed ACE test systems. Of course the small incriminating **offline code** (b1) that

⁵⁹ To put this step in perspective with respect to the entire The Yield Book valuation process, once the Cash Flow, the expected interest and principle payments from the mortgages from the prepayment model which generated the cash flow using the set of interest rate paths generated by the sequence, is discounted by those same interest rates whose accuracy depends on the sequences, the cash flow generated by all interest rate paths is averaged together to get the value of the fixed income security.

⁶⁰ It could be just **REDACTED** see the code reading in ACE term structure in V.1.21 of pathgen_cover. Term structures are the set of interest paths generated by the sequence used by The Yield Book.

reads in ACE sequences was not produced and not made a part of the RCS code, so when Russell was asked if Defendants could switch sequences using **REDACTED**

(March 2, 2006,

Russell Tr: P.173:22-174:6)⁶¹, but then admitted that “

REDACTED

” (Id, 94:7-12)⁶². In

other words, he admitted there was **offline code** and did not deny that it contained **REDACTED** that allowed the switching of sequences when asked.

F2. Right after the ACE theft, Defendants started installing new ACE Use Code in The Yield Book exclusively for Yield Book internal users including their own traders

194. Besides the major revisions in the code file `pathgen_cover.c,v` that clearly converted Theft Code into ACE Use Code for the Yield Book production right after the last ACE test on June 3, 1998, I have also found new massive ACE Use Code installed at the same time for Defendants’ YB **internal users** including their own traders.

195. The four stolen ACE sequences, one 32 paths, two 64 paths, and one 128 paths have different levels of accuracy. The Yield Book **internal users**, who including all Defendants’ traders were provided with ACE sequences right after June 16, 1998, according to The Yield Book. Clearly Defendants would prefer that their own traders have an edge over their counter parties, who are often The Yield Book **external users**, the “buy-side” institutional clients.⁶³ Therefore their own traders were given the benefit of all the ACE sequences, 32 paths, 64, and 128 paths.

196. Recall that after revision of **REDACTED** to convert the ACE Theft Code to the ACE Use Code, **REDACTED**

V19.1000.1.1., Ex. JJJJ, which was installed anew on June 16, 1998), were added to The Yield Book production only after removing the restriction that these functions could only be used by the people reference by **REDACTED** (if not Yield Book, the **inner circle** people). The remaining pieces of code **REDACTED**

mostly designed for Theft Code and were not suitable for direct use of the ACE sequences in production.

197. **REDACTED** as discussed above, Defendants also created two new suites of ACE Use Code to accommodate the needs of their own traders to use ACE in production.

⁶¹

REDACTED

⁶² **REDACTED**

⁶³ “Today, Salomon Smith Barney fixed income professionals, along with top-tier Buy Side organizations, are committed to the Yield Book as their primary analytic tool...” See , p.1 of Ex. 4 to the Declaration of Nathaniel E. Jedrey In Support of Defendants’ Motion for Summary Judgment dated April 5, 2013 (“Jedrey Decl.”).

198. First, in v1.23 of July 23, 1998 Ex. HHHH, written barely two weeks after v1.22 of July 9, 1998 (Ex. VVV), Defendants shifted the ACE Use Code in **REDACTED** from the function-oriented programming that originated in the Theft Code into a more structure-oriented programming, which is more easily debugged, in production and which is also more robust for production purposes.

199. In function-oriented programming, the programmer construct the code according the functions computer will perform. As long as the Theft Code (Fig.2, Ex. SSS dotted line actions) performed all the functions and did them one time only to save and get ACE on the standalone test computer, the robustness of the code was not a concern.

200. But once the task to “save-and-get” ACE was finished and they had stolen ACE, the focus shifted to the objects acted upon by the functions – the stolen sequences and interest rate paths generated by the stolen sequences. In the new object-oriented code, **REDACTED**

201. When I tried to find the definition of **REDACTED**,

another ACE Use Code file discussed above.

202. In **REDACTED** that appeared to be designed for the inner circle who maintain The Yield Book online production: **REDACTED**

203. In online xfiles.c, there are also parallel functions which appear to be for offline support. These functions allow the upload of the updated xfiles and contain a function save_xfiles() which calls function store_xpaths(), which calls write_paths() to saved xfile and the **offline** files are analogous to the **online** files as shown below:

Offline	Corresponds:	Online
----------------	--------------	---------------

64

REDACTED

⁶⁵ Structures are called a “struct” in C programming language.

Use_save_xpaths()		search()
↓		↓
save_xfiles()		fetch_xfiles()
↓		↓
store_xpaths()		retrieve_xpaths()
↓		↓
write_paths()		read_paths()

where the vertical arrows mean “function calls”.

204. I wanted to determine what these `xfiles` are. Mr. Russell referred Dr. Wang’s ACE sequences as “Xiaolu’s files”, (Wang Decl.). This is very similar to “xfiles” Even giving Defendants every benefit of the doubt, the evidence below leads to the inescapable conclusion that the interest rate paths in `xfiles` are generated by ACE sequences. It would be much faster to use a sequence to generate the interest rate paths used in memory, in RAM, than to read in the interest rate paths into from a data file. And, the only sequences Defendants could not generate in RAM were the stolen ACE sequences because Defendants were never given the algorithm to generate the ACE sequences in their code files.

205. Again, I looked for possible innocent explanations for this large new block of “save-and retrieve xfiles” code, but there were none. I considered any remote possibility that Defendants might want to back up their interest rate path files. However, the RCS code does not use the interest rate paths installed in PathMem contained in `xfiles` as a backup. It uses them as the default interest rate paths replacing their **online code** generated interest rate paths. It is only after both function calls `fetch_xfiles()` and `get_paths()` fail to find interest rate paths that the computer would go to `pathgen()` which generates the interest rate paths using Defendants’ own Monte Carlo code rather than reading the interest rate paths in from the data `xfiles` as shown by the following The Yield Book code in `pathgen_cover.c`, V1.23, Ex. HHHH which has remained in the code through the most recent V1.38 produced:

```

if (
#ifdef YB
    fetch_xfiles(&pid, &pm, read_paths) &&
/* TRUE if cannot get paths */
#endif
    !get_paths(&pm)) { /* returns TRUE if we can get

```



```
paths */
    status = PATHGEN(...)
```

206. As discussed above about C programming, the computer will check the criterion inside () after if () to see if it is zero (FALSE) or nonzero (TRUE). If zero, the computer will skip the code in the brackets {} following the (). And if nonzero, will go into the {} and run the code in the {}. The symbol && stands for “and.” Only if both fetch_xfiles() fails, returns a value of 1, and get_paths() fails returning the value zero, will the computer go into the {} and run pathgen(), Defendants’ interest rate generation model using their Monte Carlo code. See the definition of get_paths() in pathgen_cover.c and fetch_xfiles() below in xfiles.c, Ex. IIII:

```
static Boolean /* TRUE if paths obtained here */
get_paths(PathMem *pm)
and
int /* nonzero if not successful */
fetch_xfiles(
    PathID const *id,
    PathMem *pm,
    int /* nonzero if error */
read_paths(
    char const *path,
    PathMem *pm /* memory where paths are stored */
```

207. Here xfiles have two components: PathMem which contains the ACE term structures and PathID which contains all the market and model information used to generate PathMem from ACE sequences. The ACE Use Code hides PathID from all users of Yield Book to prevent them from backing out ACE sequences. The Yield Book **internal users** only need to use PathMem for their production.

208. Since PathMem is a nondescriptive name and programmers need to be aware of where the ACE Interest Rate Paths were, Defendants used xfiles to represent the data files holding PathMem and PathID, and using functions like fetch_xfiles() and store_xfiles, see xfile.c. Ex. IIII. We will follow Defendants’ naming convention to use the name xfile to represent the multiple files containing ACE term structures generated by one ACE sequence and held by PathMem. Defendants also used xpaths to represent the data files holding only ACE term structures PathMem, so we may use xpaths as well.

209. The function get_paths() pre-theft is shown above⁶⁶. So now, post-theft, the

⁶⁶Where the struct PathMem, *pm contains paryc and all the other arguments of the function get_paths() as its component, see xfiles.h:

```
#endif /* #ifndef YB */
typedefstruct {
    intnoutcrv;
```

get_Paths() function becomes shorter and neater, as shown above, when called by pathgen_cover()⁶⁷. When The Yield Book runs Defendants' own Monte Carlo code to get the interest rate paths paryc, the interest rate paths are automatically loaded into the computer memory, and stay there as long as The Yield Book is running, unless they are deliberately discarded (or washed out by the use of xfiles or get_paths() as is indicated above). This is specified by term "static" before the variable of paryc, in the following declaration of paryc:

```

/*
    pathgen_cover    Purpose: provides interface between pathgen
and fortran callers
*/
int
pathgen_cover(
    .....
    static double **paryc[MAXOUTCRV];
    static double **dyc[MAXOUTCRV];

```

The declaration "static" means paryc stays forever in the computer memory unless it is deliberately discarded.

210. Defendants would not wash out their **REDACTED** in the current memory just to replace it by the **REDACTED** It would waste time to no benefit. The only reason to washout the **REDACTED**

It takes hundreds times longer for the program to read in **REDACTED** into RAM from a data file than to generate them in RAM by using the **REDACTED** in The Yield Book code.

```

intconst *time_dim;
intnspaths;
intnpaths;
double ***dyc; /*[noutcrv][time_dim[ioutcrv]][nspaths]
*/
double ***paryc; /*
[noutcrv][time_dim[ioutcrv]][npaths] */
} PathMem;

```

⁶⁷ Pathgen_cover.c shows the calling of get_paths with the struct static Boolean /* TRUE if paths obtained here */ get_paths(PathMem *pm

⁶⁸ In support file history irdb.tar/history\8526.req(8) and (24):D (CGM00177) (Ex. GGGG) the support person states "I inserted the latest version from ybcomp2 into cvs and added crontab_load.sh. There are many changes for same-day Index calculations and a few changes for get_index_file and get_rover_file(a little less frequent)." (emphasis added)

211. Yet, if Defendants were using only their own sequences, it would be unthinkable for them to force their supercomputer to follow these seemingly purposeless save-and-retrieve-replACE Interest Rate Paths steps using the `xfiles`, every day and even intra-day.⁶⁸ But if they were using interest rate paths generated by ACE, it makes perfect sense because they would be forced to read something into computer memory from disk to use ACE, either the ACE sequences themselves or the interest rate paths generated by ACE because they did not have the algorithm for generating ACE in memory. They chose to generate the ACE Interest Rate Paths **offline**, where only the **inner circle** could access ACE, and read in only the ACE Interest Rate Paths **online** so users limited to **online** access could not get their hands on ACE. This was accomplished with the `xfiles`. The name and path location of the directory of `xfiles` are stored in the variable called "`index_file`" (see `xfile.c` from CGM00176, Ex. IIII).

F3. One month after ACE theft, Defendants installed the ACE Use Code for their trading and production in global financial markets

212. Defendants traded in global markets in various currencies. While in the U.S., the index curves, namely the standard yield curves that all other U.S. bonds are compared with, are treasury curves and swap curves, there are other index curves traded in other global markets. On July 24, 1998, ACE Interest Rate Paths generated using global index curves instead of U.S. Treasury and Swap curves, were installed for use in Defendants' global markets trading activities. v1.23 of **REDACTED** : July 23, 1998, login comments: **REDACTED**

REDACTED

213. There are many commands in **REDACTED** and **REDACTED** to indicate that the generation of ACE Interest Rate Paths were accomplished by the ACE Use Code file **REDACTED**. I will not cite them all for lack of space. Because of the large number of index curves used by the Defendants' global markets traders, Defendants created another code file, **REDACTED**. Just like **REDACTED**

⁶⁸ In support file history `irdb.tar\history\8526.req(8)` and `(24):D (CGM00177)` (Ex. GGGG) the support person states "I inserted the latest version from `ybcomp2` into `cvs` and added `crontab_load.sh`. There are many changes for same-day Index calculations and a few changes for `get_index_file` and `get_rover_file(a little less frequent)`." (emphasis added)

REDACTED

214. It appears that **REDACTED** have been used extensively for trading. **REDACTED**

215. The overwhelming evidence inescapably shows that right after the last ACE test on June 3, 1998, Defendants immediately installed the stolen ACE sequences into The Yield Book production for their **internal users**, i.e. employees of Defendants or their affiliates,, including all their own traders, while The Yield Book **external users** are only able to use ACE derivative sequences. See the flow chart illustrating this in *Fig. 3*. Ex. SSS.

216. The fact that Defendants withheld key Mortgage Research RCS code files ("MR-RCS") and TYB-RCS (The Yield Book production RCS Code) code files supports the conclusion that Defendants claim that they did not use ACE was false. We saw above that the RCS files produced provides conclusive evidence that Defendants' traders started using ACE term structures in two weeks after the last ACE test on June 3, 1998. Implementation of this scheme would like be through two separate sets of computer servers. For The Yield Book **external users**, Defendants would use the same online The Yield Book applications in different servers to generate interest rate paths by their Monte Carlo code using `pathgen()`. For Defendants' **internal users** and traders, i.e., the Yield Book registered clients, Defendants would use internal corporate servers which have stored hidden ACE term structures via an environmental variable. Certainly, Defendants did not go to all the trouble to steal and hide the use of ACE to provide its benefits to Defendants non-Citigroup clients and thus undercut its Traders opportunity to do arbitrage trades with such clients. The only conclusion one can reach from the factual picture painted by the RCS code produced is that Citigroup's Traders were provided with ACE and their other clients were given an inferior sequence.

217. For The Yield Book **external users**, at the same time as Citigroup traders were using ACE, **external users** would be logged into a general server where The Yield Book will not find any hidden ACE term interest rate paths in the **offline** files, and have to use the interest rate paths generated by Defendants' Monte Carlo code using an ACE derivative sequence such as LDS200, which would be implemented by **REDACTED**

We would need The Yield Book production code, including TYB-RCS and deposition of Defendants' The Yield Book personnel who has personal knowledge of installation of the interest rate paths in production for further detailed finding regarding the scope of usage of the ACE sequences and ACE derivative sequences.

218. This conclusion, that the Yield Book **internal users**, including defendants' own traders, were provided interest rate paths generated by ACE while other clients were given inferior sequences, is further supported by the following acts of Defendants:

a. From day one, Defendants' deceived Dr. Wang about their true intentions about licensing ACE because they started installing the Theft Code the day before they contact him to

ask him to test ACE in The Yield Book as shown in Section E;

- b. producing phony sequences in this litigation, Sections B and C;
- c. withholding from production ACE sequences and producing the code to cover up their use of ACE as shown in Sections B and C;
- d. refusing to provide a sworn statement by a person with knowledge of sequences they actually used in violation of the Court order while providing a deceptive declaration that purported to be in compliance as shown in Section B;
- e. falsification of ACE test results both before and after this litigation as shown in Section A;
- f. fabrication, tampering and withholding sequence development and sequence testing files while destroying the original files during this litigation as shown in Section D;
- g. implementing a hedging capability for production in The Yield Book code by adding hedge .c beginning just prior to stealing ACE as shown in Section C above, and
- h. Right after stealing ACE, changing their **online code** to use interest rate paths generated **offline** by ACE as shown in this Section F.

F4. Right after ACE theft, Defendants started installing new ACE Use Code to use ACE term structures for The Yield Book in production.

219. Following the missing link of functions, like REDACTED that were not defined in the incomplete MR-RCS code produced, I found a suite of new code files in the incomplete TYB-RCS code produced that was created right after the ACE theft and which belong to the ACE Use Code used in production. Here I only discuss one of the new code files relating to V1.22 of REDACTED which was created anew on 1998.06.16, Ex. JJJJ, V19.1000.1.1. Its first incarnation V19.1000.1.1. is only a rudimentary version. It had an identical version V20.1, also installed on 1998.06.16 (Ex. KKKK). The idea seems to be that V20 series would allow production to proceed undisturbed while Defendants worked on revising V19 series.

220. V19.1000.1.1 appears to be a dry run test. Then, in V19.1000.1.2, they got some data to test the waters before ACE term structures were used in production. This code was then transferred into V20.2 for production (Ex. LLLL). The name "REDACTED" suggests that the ACE term structures were used for computing OAS, i.e., Option Adjusted Spreads, for CMO's. The later versions soon expanded to instruments other than CMO's, i.e., ARM's, MBS's. The name hash indicates hash functions that are used to verify the data's integrity after the data has been transmitted or stored and then read back from storage. Reading in ACE Interest Rate Paths presented another potential problem for Defendants when they switched from using the term structures already in the computer memory that were generated by their Monte Carlo code online, to ACE term structures read in from data files stored by offline.

221. To store and retrieve ACE term structure contained in large data files of gigabyte size may cause mechanical errors in hard disks that result in an inaccurate transfer of the ACE Interest Rate Paths. Such errors if not detected and corrected, may cause large errors in pricing and trading. Defendants' installation of hash functions solved this problem. How do they know the retrieved file was likely correct? They used a hash function which generates a smaller file that is sent together with the large term structure file. To illustrate the idea of a hash function,

let's use an extremely simple example. Suppose the term structure has only say, 3 numbers, 1.0, 2.0, and 3.0 (%), stored and then retrieved. The hash function is, say, the addition of all the numbers, $1+2+3=6.0$. Then the small output file containing "6.0" is stored and retrieved with the original file containing 1, 2 and 3. They run hash function again after retrieval and to match the output with the stored hash file. If they still get 6.0, then there is unlikely to be any error. If they get any other number like 6.5, then they know that an error occurred in "store and retrieve", and the file will be removed. Then they would simply retrieve it again.

222. Most of the code in REDACTED (Exs. JJJJ, KKKK, LLLL) included in the key ACE Use Code deals with REDACTED v1.22 of 1998.06.23. The function REDACTED¹ v20.2 (which is identical to V19.1000.1.2) of 1998.06.25, and all later versions of REDACTED conv the ACE REDACTED for The Yield Book production⁷⁰. The same REDACTED is used in V1.22 of REDACTED, Ex. VVV. The two functions REDACTED and REDACTED REDACTED But the definition of the two functions cannot be found anywhere in MR-RCS nor in TYB-RCS produced by Defendants because Defendants have removed from CGM00177 all code files relating to term structure generation.

F5. Right after the ACE theft, Defendants started compiling executable code using the ACE Use Code to build The Yield Book systems for The Yield Book production

223. The RCS compiler scripts, including REDACTED compiled the aforementioned ACE Use Code into libraries of object code⁷⁰. As shown by for MR-RCS, The Yield Book code was recompiled after the preprocessor REDACTED on the ACE Theft Code in REDACTED to transform it into ACE Use Code, on June 16, 1998. See ¶¶175-176 above. When a new suite of ACE Use Code centered around REDACTED was created on July 23, 1998, the script file REDACTED Ex.EEEEE, indicated a new revision v1.11 REDACTED. REDACTED 1.11 1998/07/23 21:06:19 russell Exp russell \$log comment reads

69

REDACTED
REDACTED

REDACTED The process of building an executable has at least two steps: first compile all the source code to build a library of object code, using script file "makelist"; Then link all the library object using a script "makefile".

⁷⁰ The process of building an executable has at least two steps: first compile all the source code to build a library of object code, using script file "makelist"; Then link all the library object using a script "makefile".

REDACTED Other scripts REDACTED

All these applications use object code libraries for term structures. This shows when Defendants started compiling and building executable code using ACE Use Code for The Yield Book production. All these files were produced as part of CGM00176.

F6. Right after ACE theft, Defendants had to install Locking Software to avoid conflicts between The Yield Book Online System Downloading ACE Term Structures and The Yield Book Offline System Uploading ACE Term Structures

224. For the theft operation to save the ACE sequences - "save_seq", Defendants could "save_paths()" during ACE tests, and "get_paths()" afterwards without any risk of disclosure of the ACE sequences to a third party or non-complicit employees because Russell was the only operator using both the save and get functions in a standalone computer. However, Defendants did not want to use the stolen ACE sequences in a manner that would allow a third party or an employee that they did not trust with the stolen ACE sequences to have access to the ACE sequences. Using the ACE in a secretive manner in The Yield Book to prevent such access represented a considerable programming challenge. If Defendants had just put the ACE sequences directly **online** and generated interest rate paths, they would run the risk that all The Yield Book group employees with access to their files could directly access ACE and copy it. Having stolen it in the first place, Defendants would hardly be in a position to complain. Accordingly, they wanted to limit access to a select group of people, the **inner circle** people.

225. So instead of designing the ACE Use Code to read in ACE directly as was done in the ACE Test System (Fig.2, Ex. SSS), the ACE Use Code was designed to use ACE to generate the interest rate paths **offline**, save them into PathMem, or an xfile, using the code xfiles.c etc, and then only put the ACE Interest Rate Paths contained in PathMem **online** for The Yield Book production. This is shown by v1.24 of July 24, 1998 has login comment: "@hide PathID from Yield Book". This had two advantages. First, if anyone copied the PathMem containing the ACE Interest Rate Paths, they would not be able to back out the ACE sequences because they would not know PathID, which contains the exact interest rate model parameter settings used by Defendants when the **offline code** in xfiles.c or pathgen_cover.c was run to generate the interest rate paths to PathMem. Second, it helped to conceal the use of the stolen ACE sequences making it very difficult for plaintiff's experts to discover what they were doing. Defendants had to update their ACE Interest Rate Paths throughout the day, every day. Because the ACE interest rates had to be saved to a data file **offline**, this was burdensome. But this was a burden they were willing to endure to gain the extra security.

226. This is another clear indication to Defendants' recognition of the immense value provided by the accuracy and speed of ACE. Otherwise Defendants would not have taken such extreme measures to steal the ACE sequences and to ensure their security.

227. However, this security measure of putting only REDACTED, the ACE Interest Rate Paths, **online** instead of ACE **online**, created another risk, the risk that the **offline** application updating REDACTED or REDACTED may conflict with an **online** application reading the same REDACTED run by The Yield Book users. Such a conflict could cause an erroneous reading of the and result in disastrous pricing results or crashing their supercomputer.

228. Moreover, there was a very big risk of this conflict occurring on a regular basis. It takes much longer to read into the **offline** code the current yields needed to generate the interest rate paths and then generate the rate paths with a sequence for all the time steps than to read in to the **online** code just the ACE sequence to generate those interest rate paths. Yet Defendants had to do this if they were going to generate the interest rate paths **offline** to hide the use of ACE. . Thus updating the interest rate paths takes a significant amount of time. And the interest rate paths need to be updated daily or even intra-day. If the **online** files tried to read interest rate paths for the prepayment model to value securities while this updating process was taking place, there would be a conflict.

229. Defendants implemented a clever solution to this conflict problem - a solution used for the first time ever used in The Yield Book. The solution uses a software function named REDACTED developed by BSD (Berkeley Software Distribution). Defendants' **offline** The Yield Book executable⁷¹ REDACTED when it starts to REDACTED It then REDACTED in particular the other The Yield Book **online** executable, REDACTED The same REDACTED is activated by The Yield Book **online** executable, using a REDACTED whenever The Yield Book **online** executable is REDACTED . This process makes the REDACTED either by the **offline** The Yield Book executable, or by REDACTED until the **online** The Yield Book REDACTED for trading.

230. If the REDACTED operation were just a backup operation, the conflict would not have occurred and no REDACTED would have been necessary because the computer is always multitasking and programming code can always be multi-threaded. If the REDACTED were a backup, the REDACTED REDACTED would already be in the computer memory and the supercomputer can feed traders REDACTED from its memory while saving REDACTED to a disk file without any risk of a conflict. Multiple **online** users may use the same REDACTED without conflict, because they are running different threads of the same The Yield Book **online** executable and share the same REDACTED in the same memory.

231. For the same reason, if Defendants used the sequence generator in their The Yield

An executable is a program that can be launched stand alone in a computer, such as by a click of a mouse in a PC.⁷² According to "A Term Structure Model" of Defendants, it took a minute for calibration of the term structure model (p.16 fn 7, Ex. S), so it would be seconds for generating the interest rate paths of 200 paths in 1997. Writing them onto a disk file would take much longer.⁷³ REDACTED

REDACTED

Book code to generate interest rate paths, then there would be no conflict, because there would be only one The Yield Book executable running. The same The Yield Book **online** executable would be multi-threading - it automatically lines up all tasks involving the rate paths into a queue and it has the algorithm for generating the **online** sequence, the sequence generator, and could generate the interest rate paths very quickly⁷² in the RAM memory for pricing.

232. For the same reason, if the traders using the online The Yield Book could have access ACE directly in the RCS code, read it, and use it to generate rate paths, that conflict would not have arisen.

233. The conflict arose from the fact that, (a) it is not a backup operation, The Yield Book relied on ^{REDACTED} for trading as discussed above; and (b) the traders and the support programmer who was updating ^{REDACTED} are two different groups of users who might access the same ^{REDACTED} ; at the same time.

234. Thus this code using the ^{REDACTED} demonstrates that there are two types of separate executable programs using two separate sets of computer memories. One executable cannot talk directly with the other, even though they may be running simultaneously. If they try to access the same interest rate path file, a conflict will result as discussed above.

235. The first executable program is the executable that is online and used by The Yield Book trader to use the ^{REDACTED} the file containing the ACE Interest Rate Paths, for pricing. It reads the ^{REDACTED} into the computer memory where it is then used to price MBS. The RCS code shows that right after the ACE theft, The Yield Book no longer used as a default the rate paths generated using the sequence generated by the sequence generator in the online code. Instead, they switched the default to read in the interest rate paths in the ^{REDACTED} Only when the online executable program fails to get the ^{REDACTED} and it fails to get saved rate paths generated by ACE, would the online executable use rate paths generated by Defendants' online sequence generator.

236. The second executable program is run offline by someone who had access to the ACE sequences and who uses them to generate the interest rate paths offline in the same computer or on a separate computer. This offline executable is separate from the first online executable, having separate memory allocation. The first program cannot access the interest rate paths in the second program's allocated memory even though the interest rate paths may share the same variable names. So when the first program tries to write the rate paths from memory (the ACE generated rate paths) to the disk file (the ^{REDACTED} while the second program is reading this same file (the ^{REDACTED} into memory, it will result in a conflict. A file cannot be read when data is being written into it. The ^{REDACTED} function neatly solved that problem.

⁷² According to "A Term Structure Model" of Defendants, it took a minute for calibration of the term structure model (p.16 fn 7, Ex. S), so it would be seconds for generating the interest rate paths of 200 paths in 1997. Writing them onto a disk file would take much longer.⁷³ / ^{REDACTED}

REDACTED

F7. Defendants promptly cleaned off no longer needed Theft Code (B9)

237. In the version 1.39, November 13, 1996, of `get_deal.c,v`, Defendants added the ACE theft code, such as the function call `use_save_paths()`, on p.19, Ex. MMMM:

```
#ifndef YB
    if(use_save_paths())flag=TRUE;/*later deal may need30-
year*/#endif
```

V1.39 also added the code to make sure that term structures all the way out for 30 years future curves will be generated⁷³,

```
#include "proto_bpamort.h" /* SET_30_YR() */
```

1. The only difference between v1.40 on November 14, 1996, Ex. NNNN, and v1.39 the day before on November 13, 1996, of the code file `get_deal.c`, is the line

```
#ifndef YB
#include fproto.h /* use_save_paths() */
#endif
```

There were more than 300 functions declared in `fproto.h`. Ex. KKK. The only function that Defendants put in the comments of `get_deal.c,v`, to remind them why they were including the use of `fproto.h` was to `/* use_save_paths() */`. At that time, Mr. Russell was still deciding where to insert code needed to call the ACE Theft Code. Up to that time, there had been no actual calls to use any ACE Theft Code, including the function `use_save_paths()` in v1.40, and v1.41 of 1996.11.21. This is consistent with my finding before that while the ACE Theft Code made its debut on August 1, 1996, it was still being revised, and was not yet ready for “production”, throughout the period when Defendants were testing ACE.

238. Defendants have not produced `cmo_api.h` which is used in all versions of `get_deal.c`, see the call: `#include "cmo_api.h"`, on p.1. In the version 1.42, 1996.11.16, of `get_deal.c,v`. Ex. BBBB. Defendants added the ACE Theft Code, such as the functions `set_save_seq()` and:

```
use_save_paths().
#ifndef YB
```

⁷³/*set flag to indicate calculation of 30-year rates also */

```
    while (indexn--) {
        if (!strcmp(deal->index_list[indexn], "TSY30")) {
            flag = TRUE;
            break;}}
#ifndef YB
    if (use_save_paths()) flag = TRUE; /* later deal may need 30-year
*/#endif
    SET_30_YR(&flag); /* TRUE if TSY30 is found */
```



```
if(use_save_paths())flag=TRUE;/*later deal may need30-
year*/#endif
```

239. This **REDACTED** .. There were more than 300 functions listed in ⁷⁶Yet, the only function that Defendants put in the comments of

REDACTED

After the successful theft of ACE in June 1998, in the mop up operation to clean up the dangling loose ends of theft code, Defendants deleted this very code line:

REDACTED

in ⁷⁵Ex. _BBBB.

F8. Defendants' admission in the ACE Use Code that The Yield Book was using "multiple sequences fewer than 200 paths", during the period 2000/10-2006/4 when they claim that "their" sequence consisted only one LDS200, besides the phony "1000" paths shows they were using ACE.

240. Defendants' code required the use of a sequence of fewer than 200 paths.⁷⁴ However, during the relevant time period from 2000 through 2005 when the RCS code was produced, Defendants claim to have been using only a single 200 path mixed seed sequence and a 1000 path single seed sequence. Neither of these sequences satisfies this requirement that it have fewer than 200 paths. As of 2005, the only sequences that Defendants had ever tested (and not previously discarded-) that met this "fewer than 200 path" requirement were the ACE 32, 64 and 128 path sequences. Defendants would never have put into production a sequence that they themselves had not thoroughly tested, as demonstrated by the extensive testing of first ACE and then Teytel's extensive testing of the mixed seed sequences. See also Report and Recommendation of August 5, 2009, p. 33 ("ACE, or any analog, would not exist without a substantial documentary record of its creation."). Accordingly, the only reasonable explanation for this code requirement of fewer than 200 paths is that Defendants' code was designed to call up, and in fact did call up, the thoroughly tested ACE sequences.

241. Also in 2005, Defendants' code elsewhere referred to the use of multiple sequences when they commented "paths varies with run", each consisting of fewer than 202 paths.⁷⁵ As of 2005, Defendants claim to have been using for production purposes only one

⁷⁴See **REDACTED** (CGM00176)(Ex. CCCC)

⁷⁵In a comment in effect from 2001- the date of the last version when the code was produced in 2006 - in file

sequence of fewer than 202 paths -- a 200 path mixed seed sequence. Ex. W, Interrogatory Responses 1-4. Thus that could not have been what Defendants were referring to when they said “paths varies with run” of fewer than 202 paths to which Defendants’ code referred could only have been the ACE 32, 64 and 128 path sequences. Again, as noted above, apart from Defendants’ 200 path mixed seed sequence, the ACE sequences were the only sequences of fewer than 202 paths that as of 2005 Defendants had ever tested and not discarded.

242. In the code, ccoas.c (Ex.CCCC), ccoas stands for current coupon option adjusted spread. Current coupon MBS, IO's and PO's are the most frequently traded instruments. Its functions are to calculate current coupon OAS (option adjusted spread) from the market prices and then save them into file: ccoas_file. When their traders price other instruments, say a CMO, they will get “oas” from the ccoas_file and compare them to get a fair “oas” for the bond. From the ccoas.c function calls one can see how they calculate all CMO's (using cmoopt compiled using maincore and other files) and all MBS, IO's and PO's (using MORTOPT) to see if they match with the current market (Ex.WW flow chart of use_ccoas_correct function calls).

243. Look at the function use_cc_correct.⁷⁶ Its value is set to be 0 (False) at the beginning. This does not change until ccoas checks the oas of the instrument the trader wants to trade and matches it with ccoas_file to confirms that it is a profitable trade. Only then will its value be changed to 1(True). This appears to be a step a trader has to go through to make a deal.

244. Because ccoas is calculated using a particular sequences, to match the oas of any of the instruments, including CMO's MBS, IO's, PO's, The Yield Book must use the same sequences that had calculated ccoas_file. Otherwise, the oas will not be able to match. There are multiple of sequences with path numbers less than 199. This is why the path number pathn needs to be printed. Based on the foregoing, we can conclude that in the calculation of ccoas for Defendants' **internal users** (who had the benefit of nts, i.e., new term structure, and this ccoas.c) used only ACE to calculate current coupon oas for all liquid instruments, including current IO's, PO's, and MBS. They also used ACE and only ACE for calculations of all the other instruments: CMO's, IO's, PO's.

Oas_deal.h,v (Ex. DDDD), it defines the maximum paths as “202” and states that the “number of paths varies with the run” to price securities:

```
#define PATHMAXC 202 /* array size only! 0=constant, 1=forward,
2+=stoch. */
/* number of stochastic paths varies with run- see read_n_factor
*/
```

```
76 Boolean use_ccs_correct( void) {
    return ccs_correct;
}
```

```
Boolean use_ccs_correct_(void) { return use_ccs_correct(); } /*
yc_paths_2.F */
```

F9. Defendants' code shows that their new term structure model in The Yield Book was *not* using their Monte Carlo code, but rather exclusively using ACE for accurate pricing.

245. If Defendants were not using the ACE sequences, the calibration of the new term structure model would have to use a sequence generated by The Yield Book Monte Carlo code, either in `gauss_random.c` or `gauss_random_mixed`. The code file `nts_monte_calibration.c`, v. Ex. T, shows that from April 8, 2002, until the last version of that code file that was produced, the calibration of the new term structure model used `monte.c`. Yet Defendants' code reminder in log comments to v1.1 creating the code file `nts_monte_calibration.c` reminded the programmer that when The Yield Book was using `monte.c` to generate term structures, it was not using their Monte Carlo code (their code for generating interest rate paths, using their random number generator.⁷⁷):

1.1.

```
date 2002.04.08.16.53.56; author rb11722; state Exp;
branches;
next ;
desc
```

@Monte Carlo code not currently used in the nts model.(emphasis added)

246. Because the sequence used to generate the interest rate paths for the calibration of the new term structures did not use the Monte Carlo code, the sequence used to generate the interest rate paths had to come from a data file. This sequence had to be a combination of ACE sequences because Defendants had no other sequences stored in data files other than ACE sequences (Section E). They only had access to ACE data files and they don't know how to generate ACE so they had to use sequence data files when using ACE.

247. This is consistent with the fact that the new term structure models were exclusively used by Defendants' **internal users** including their own traders. All of the code for their **internal users** is to read in interest rate paths from an external source. As I discussed above, Defendants' **internal users** used PathMem which contained the ACE term structures. Defendants have withheld from discovery the files they actually used to calibrate their Term Structure Model which used the so called "1000" path sequence which was in reality a SuperACE sequence.

⁷⁷ Radak 30(b)(6) Sept. 28, 2007 dep. 27:6-13: "Q. Well, to generate an interest -- to generate future interest rate scenarios to be input into the prepayment model, what would you have to do REDACTED to generate future interest rate scenarios? A. REDACTED

F10. Relying on stolen ACE, Defendants for the first time implemented arbitrage trading strategy to cherry pick mispriced MBS/CMO's and hedge them earning profits with virtually no risk and a relatively small amount of capital

248. Defendants' ACE testing showed them that ACE was an order of magnitude more accurate than their Production200, Defendants production sequence in use. Fan Notice ¶ 20 (Ex. A). Defendants massively tampered with the test results, as a part of their efforts to cover up their ACE theft, *See* Section E. Defendants admit to being market makers in MBS including illiquid MBS like CMOs. Market makers as a matter of course would want to be able to hedge their trading portfolios. However, Production200 was not accurate enough for them to hedge.

249. Defendants were well aware of the concept of arbitrage with respect to illiquid MBS when they were testing ACE. They knew all about the risk of becoming a victim of arbitrage if the interest rate paths were not accurate enough and said so in a June 1997 paper entitled "A Term Structure Model and the Pricing of Fixed-Income Securities", "Otherwise, we could become victims of arbitrage." Ex. S, p. 7 AAI 0688. In that paper, they described their desire to achieve "arbitrage free pricing" at 12, Ex. S. Defendants were well aware of the arbitrage advantage in pricing a more accurate set of interest rates would provide. The article was co-authored by Robert Russell who was in charge of testing ACE and stealing ACE.

250. Defendants first started to experiment with it REDACTED
The last version was logged in in January 1995 and
experimental only as indicated by the warning comment

REDACTED

251. Defendants installed ACE Theft code the day before they call Dr. Wang to test ACE in August 1996. *See* E. This demonstrates that at no time did Defendants deal in good faith with Dr. Wang. They planned to steal ACE if it were proven to be sufficiently accurate during testing right from the beginning.

252. Defendants' code did not permit hedging before they stole ACE. Thus, they clearly did not view their Production200 capable of supporting an arbitrage strategy that required hedging. As noted above with respect to *hedging.c,v*, Defendants at best were only experimenting with hedging until Dr. Wang arrived on the scene with ACE.

253. *hedging.c,v*, which had had only RED revisions, the last in REDACTED was quickly abandoned in 1995 when the code grew to only REDACTED REDACTED. It was a failed experiment. However, it appears that after only two rounds of ACE testing, Defendants recognized that ACE was so accurate that they could use it for hedging and arbitrage. So they started to create code to allow hedging in production. On REDACTED, less than three months before Defendants stole ACE, they installed first version of hedge code *hedge.c*, version 1.1. Version 1.1 started off with REDACTED code lines. It was REDAC times the size of the abandoned *hedging.c*. *Hedge.c* was rapidly modified REDAC times in REDACTED prior to the theft of ACE on June 3,

1998, and Defendants REDACTED A copy of the RCS version of REDACTED containing various versions of hedge.c and showing the revision dates is Ex. MM.

254. From REDACTED Defendants fixed the bugs in hedge.c to make it work for production. Finally, the log comment of REDACTED announced the success of dynamic hedging for at least MBS instruments:

REDACTED

255. From, REDACTED log comments indicated to their effort to hedge portfolios:

REDACTED

The log comments for REDACTED announced the success:

REDACTED

Here, a REDACTED is a CMO that was purchased that no longer needs to be sold. Instead, it held in a portfolio to be hedged, a portfolio REDACTED to hold its value. A REDACTED is a pool of liquid benchmark instruments that replicate the cash flow generated by CMO. These liquid instruments are sold short to "hedge" the CMO. REDACTED the same effect as selling the CMO put into the trust and hence that CMO is hedged. When the two cash flows are identical, then

REDACTED
REDACTED

256. As discussed in Section C, using a Super ACE sequence for calibration of the term structure model while using the smaller and quicker ACE constituent sequences for pricing appears to be the game changer that enabled them to successfully follow this strategy. A key issue is whether Defendants were successful in using ACE to arbitrage the illiquid MBS market. The fact that they continued the strategy from 1998 through the present time⁷⁸ is a very good indication that they were. Damage discovery is necessary to determine the extent of such profits

⁷⁸ See, Richard Isenberg dep. 8/14/2012 discussing portfolio hedging of MBS by Defendants' institutional traders, 177:24-185:5, Ex. UU and Richard Isenberg declaration dated June 18, 2012 ¶ 2 and 6 (Ex. V V) explaining that The Yield Book is a tool used by traders each day to adjust their model valuations to value individual securities and their hedges including for CMOs. This strongly suggests that Defendants are still following this arbitrage strategy first implemented when they stole ACE. See Sections C and F below.

G. Defendants' Motion to Exclude Mischaracterized the Record and My Testimony

257. I have reviewed the motion filed by Defendants on April 5, 2013 ("Motion to Exclude") to exclude my trial testimony relating to my first expert report dated October 15, 2007 ("First Report")(Ex. G hereto). It mischaracterized my First Report in many ways and relies on facts that have no evidentiary support.

258. First, Defendants say that my testimony should be excluded because my First Report "does not analyze – much less assess in a scientifically reliable way – whether Defendants' actual product [meaning sequences] bears any actionable similarity to, or was derived from, AAI's alleged trade secret, the ACE Numbers [sequences]". Motion to Exclude p. 1. As demonstrated in Sections A-F above, Defendants have not produced any of the sequences they used which include the ACE sequences. In fact they have not even produced a single sequence that would be generated from the default seeds by The Yield Book code files produced. They have produced no authentic evidence to support that any of the default seeds are real seeds for LDS100 or LDS200. They even produced one sequence, the purported Russell 1000 path sequence created by Dr. Radak, that cannot be generated at all by their production code because the code does not permit more than total paths with more than 3 digits. Thus it is not surprising that no one with first had knowledge of the actual sequences used by Defendants The Yield Book has verified under oath that the sequences produced are all the sequences used in production despite a Court order to do so.

259. In essence, Defendants seek to preclude my testimony because I have not done a comparison analysis of ACE to the phony sequences that they improperly produced in discovery. Defendants ignore the fact that my First Report concludes that Defendants had stolen and were using ACE, a conclusion that has been strengthened by additional discoveries since the First Report was rendered as demonstrated in the previous sections of this Declaration. In other words, Defendants stole ACE, then first used ACE verbatim for production purposes, and then used an algorithm to select LDS sequences by targeting stolen ACE– when required to produce in discovery the sequences they used – produced phony sequences instead of ACE. I used the first portion of LDS64 to analyze what is the real algorithm (Defendants admitted that they used the same algorithm to select all LDS sequences) because that is the only authentic portion of LDS sequences, and is the common portion of all successful LDS64 candidates. If Defendants' position were correct, then when a person who steals a trade secret is sued, if he simply does not produce the stolen trade secret in discovery, he can never be held accountable. Such a rule would shield theft.

260. Defendants point out that Dr. Teytel testified unequivocally that all LDS sequences, including LDS64, LDS100, and LDS200 - which they claim are "Defendants' actual sequences", were selected using one and the same algorithm ("Real Algorithm"). The only relevant question is what is the Real Algorithm used by Dr. Teytel. As the evidence presented in my first Expert Report concludes, the Real Algorithm is not Teytel's Mixed Seed Algorithm, but

an algorithm targeting ACE. Ex. G.

261. The “Mixed Seed Algorithm” authored by Dr. Teytel makes no mathematical sense. The Motion to Exclude claimed that there is no basis for me to use correlation analysis. However, it was none other than Dr. Teytel himself who raised the idea of using “correlation analysis” that allowed me to discover the Real Algorithm. Teytel justified the Mixed Seed Algorithm by so called “correlations between two portfolios” which he purportedly calculated:

REDACTED

Dr. Teytel has never presented this “analysis” or any data supporting this “analysis” anywhere. He could not because a **REDACTED** can never even be defined mathematically, whether the **REDACTED** are large or small. Even Teytel’s boss, Dr. Lakhbir Hayre, admitted that **REDACTED** makes no sense, *see* Ex. G, Fan Expert Report at p. 9 and fn xvi quoting Dr. Hayre.

262. The Motion to Exclude stated that Fan 2007 Report did not cite any reference for my standard statistical analysis. The method used in my analysis is the most basic theory of statistical decision or otherwise known as theory of hypothesis testing. I have taught the subject for many years and saw no need to cite such standard textbook. Dr. Hayre immediately knew what it was as shown above.

263. A standard analysis starts by formulating a hypothesis, known as “null hypothesis”, to test its validity. In our case, the null hypothesis is “Teytel developed LDS without relying on ACE”. I used the population of the first segment of 8 dimensions (a sequence of 16 numbers as there are two factors in each dimension) of the 64 paths of LDS64 generated by Best Seed as the sample to compare its approximation to ACE64 with that of the general population of all candidates LDS64. The result is that from the sample of the first 8 dimensions, our null hypothesis must be rejected with level of significance of 0.005, i.e., with confidence level of 99.5%. This is elementary statistics.

264. The motion to exclude my Expert Report dated October 15, 2007, stated that the term “significant” is undefined when I used the term “significant” for the approximation of Teytel’s LDS64 to ACE64. However, “significant” is a standard term used in all statistical analysis in science, including social science, engineering, and medicine..., when the null hypothesis is rejected with a level of confidence (for making a false rejection) is 0.05 or less. In our case, the level of confidence is 0.005, far less than the commonly accepted 0.05. Thus, it is very clear that Dr. Teytel was targeting ACE64.

265. The second basis for striking my testimony is that much of my first Expert Report was based upon work done by Dr. Sen Hu, an individual who resides in China and is unavailable to Defendants, and that supposedly blind reliance on Dr. Hu’s work is inappropriate. First of all, in my view, there is nothing wrong with relying on Dr. Hu’s work. He is a superb mathematician

with world class credentials including Professor of Mathematics and Director of the Institute of Mathematics at the University of Science and Technology of China, and has a Ph.D. from Princeton University (where I am a chaired professor). He was also an assistant professor at Northwestern University. I know the quality of his credentials, and I am confident of his expertise. Further I have discussed the issues with him extensively. In my opinion, relying on work he did is wholly appropriate. In addition, I have since checked all his work that I had not yet checked in 2007 and reviewed all the code he used to generate the sequence I relied upon in my first Expert Report, and independently verified the results used in my previous reports.

266. But Defendants are not forced to rely on my or Dr. Hu's calculations. Defendants had sufficient information to duplicate the work Dr. Hu upon which I relied. Defendants knew which of Defendants' seeds Dr. Hu and I used because they are disclosed in the First Report and Dr. Hu's Fourth Declaration; they knew that the code used to generate the sequences from the seeds identified was their own `gauss_random.c` produced to Plaintiff. I testified to that effect at my deposition. *Fan dep. 12/11/07 95:7-13 - Q. How did you generate the sequence from the seed referenced on Ex. as 13812? A. That was generated by Dr. Hu and to my best knowledge he ran -- he used a program and I think gauss_random.C putting the seed there.*" Using `gauss_random.c`, **REDACTED** to generate the first portion of LDS64 using the "Best seed" is a standard procedure⁷⁹ (I verified the code and generated the sequence myself later). Generating LDS64 candidates with 100,000 more different seeds, is also a standard procedure. Defendants or their experts could have generated the LDS64 and also 100,000 LDS64 candidates themselves, just as I did, and compared their results to mine. They chose not to have their own expert try to duplicate either Dr. Hu's tests or my tests. The results and analysis of the results are accurate. Defendants can hardly be heard to dispute those results now, having not bothered to run the tests themselves. (Or, perhaps, having run the tests themselves, Defendants found that Dr. Hu and I were indeed correct, but Defendants and their expert chose not to reveal that finding.)

267. Defendants also mischaracterized my finding as just one seed out of hundreds of thousands seeds had similarity to ACE and questioned why only just one portion of LDS64 was used. Dr. Teytel, not me, selected the "Best" seed 13812 ("Best Seed") from a very large number of seeds. He decided that the **REDACTED** determined by the Best Seed is the best. And then he tested hundreds of thousands seeds **REDACTED** of LDS64, but they **REDACTED** the Best Seeds. So the only authentic portion of any LDS sequences is the first common segment of all the LDS64 sequences generated using the same seed 13812, the only Best Seed identified in the Teytel Notebook (CGM00209)(Ex. SS) among hundreds of thousands of seed candidates reflected in his Notebook as being tested by Dr. Teytel .

268. Using the same code used by Dr. Teytel, I created 100,000 candidate LDS64 sequences using 100,000 different seeds, rather than the Best Seed which determines LDS64. I then matched the first 8 dimensions of each of the 100,000 candidate LDS64 sequences with ACE 64 sequence, and computed their average correlations for the best 16 and 32 matched paths. Hence, the 16-paths and 32-paths of the LDS64 determined by the "Best Seed" approximate the

⁷⁹ Radak Code produced Radak Sequences using 3 code files with similar names but inserted bogus path numbers, bogus dimensions and bogus seeds not in The Yield Book code, and improperly shuffled factors.

matched ACE paths significantly better than the general candidate LDS64 population.

269. Comparison of this first segment of LDS64 determined by the Best Seed, to the first segment of ACE64 demonstrated that it was selected by targeting ACE with a 99.5% certainty (the probability of such a strong similarity occurred by chance is only 5 in 1000). The Distributions of the LDS Sequences generated by the Best Seed selected by Dr. Teytel are significantly similar to the distributions of ACE horizontally on most of the dimensions, even when the distributions are not the bell curve distribution.

270. Motion to Exclude p. 7 citing Fan TR. 263:21-25. Defendants challenge my conclusion without the benefit of an expert opinion or any apparent undertaking to duplicate my test. Contrary to Defendants' inept, bald assertion, the data from eight dimensions (16 numbers altogether from two factors) times 64 paths that I used was an ample sample size to support a reliable conclusion. The fact that a comparison of more dimensions would have resulted in a lower absolute correlation does not in any way invalidate my conclusion, as Dr. Teytel used the first seed to target only the **REDACTED**. Defendants simply do not understand my analysis or are intentionally trying to confuse the Court. The conclusion I reached, based on standard mathematical analysis, is that the correlation between LDS64 and ACE64 is so high that there is a 99.5% probability that the LDS64 generated by the "Best" seed, 13812 was found by an algorithm targeting ACE. We find that the best seeds are indeed best in terms of the approximation of the sequence determined by the best to the ACE sequence both horizontally and vertically.

271. Next, Defendants try to undercut my conclusion concerning the comparison of ACE64 to the Best seed by highlighting testimony where I admitted inconsequential errors in Ex. s supporting my opinion. Motion to Exclude p. 7 citing Fan TR. 272:06-273:15; 283:07-284:23. As Defendants well know, that testimony pertained to the Ex. s before I submitted the corrected charts all of which support my conclusion. Fan Dep. 272:3-273:6; 283:7-303:11. As explained above, Defendants had all the information they needed to duplicate and challenge my conclusions in the more than five years that have passed since that deposition. They have chosen not to do so. I stand by my opinion.

272. As for Defendants' claim that we lack sufficient expertise in computer code, both Dr. Hu and I have extensive experience in writing computer code that is more than sufficient to support the opinions rendered. Defendants have offered no opinion by an expert to contradict that fact. As to whether Dr. Hu is available for deposition, I really cannot say whether he is. But I was available and was deposed for two days on the subject of my first Expert Report, and I am available to stand for deposition on this declaration.

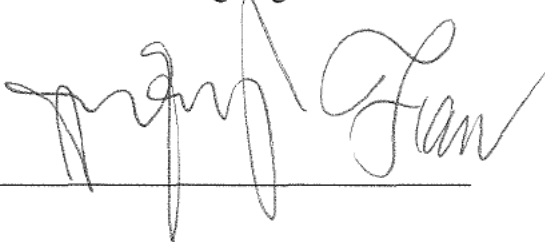
273. In sum, my First Report is based on solid evidence directed at a key issue in this case whether Defendants stole and used ACE. Moreover, its conclusions have been further supported by additional evidence that Defendants stole and used ACE to implement, for the first time, an arbitrage opportunity that they were aware of prior to the theft of ACE but could not implement because their Production200 sequence was not accurate enough to enable them to construct effective hedges.

H. Defendants withheld The Yield Book production code, TYB-RCS and CMO-RCS, including the offline code read in ACE sequences

274. Defendants withheld from production *all* of the term structure code of The Yield Book group's RCS code ("TYB-RCS Code") and produced only portions of the Mortgage Research group's RCS code ("MR-RCS Code"), see D4, and only for the period before July 2005, and withheld all the code from their CMO Analytics Group ("CMO-RCS") as disclosed by Defendants on July 13, 2012. They withheld the small but inculpatary **offline code** reading in ACE. It is undisputed that Defendants used The Yield Book, in particular, cmoopt to price CMO's to test sequences. The application cmoopt uses the cash flow generator produced by CMO-RCS, as confirmed by Teytel declaration of July 13, 2012 and the Figs. 1-3 of Ex.SSS. The vast majority of the fabricated Development Production files are purported outputs of cmoopt. Although Defendants misrepresented the outputs as used by Teytel to select seeds, but they appear to be outputs of cmoopt to calculate benchmarks for testing sequences. So TYB-RCS and CMO-RCS are part of the sequence development files and testing files, that have not been destroyed by Defendants, when they destroyed all the original seed selection code. Defendants have produced no test systems for any sequences. Given all the evidence previously concealed with respect to the phony Radak Sequences, the fabricated Production, I will need the complete TYB-RCS Code and the complete MR-RCS Code, and the CMO-RCS Code, and depositions of witness from Defendants' **inner circle**, Mr. Robert Russell and Mr. Stewart Herman who created the Theft and Use Code, to determine when and how ACE sequences were used by The Yield Book.

Dated: June 4, 2013

I, Jianqing Fan, declare under the penalty of perjury under the laws of the United States of America that the foregoing is true and accurate.

A handwritten signature in black ink, appearing to read "Jianqing Fan", is written over a horizontal line.